# TECHNICAL MEMORANDUM

WL-TM-91-129

## EARLY BIRD FINAL REPORT

30 DECEMBER 1991

MICHAEL L. BOHLER
CAPT F. JESSE FANNING
JEFFERY S. SIFERD
BARBARA L. ELDRIDGE
CAPT RAFAEL MORALES
LT GREGORY A. WEBER



Approved for public release;
distribution is unlimited.

WRIGHT LABORATORY
AVIONICS DIRECTORATE
SYSTEM INTEGRATION BRANCH
WRIGHT-PATTERSON AFB, OH 45433



# Early Bird Final Report

**Michael L. Bohler** **Barbara L. Eldridge**

**Capt F. Jesse Fanning** **Capt Rafael Morales**

**Jeffery S. Siferd** **Lt Gregory A. Weber**

**Wright Laboratory**
**Avionics Directorate**
**System Integration Branch**
**Wright-Patterson AFB, OH**

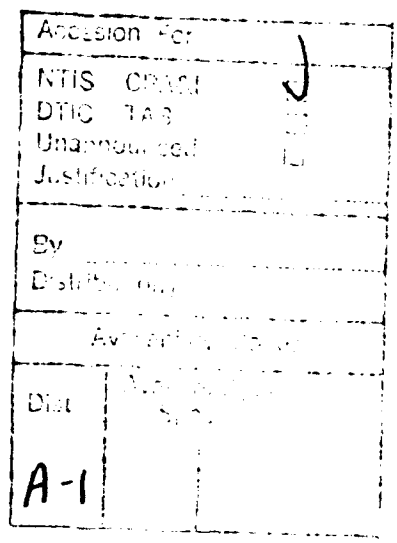


**Table of Contents**

## Appendix Table of Contents

## 1.0 Early Bird Program Introduction

1.1 Integrated Systems and Flight Test. In order to combat a numerically superior opponent in a battle, integration of available resources is necessary. In particular, by improving the capabilities of each platform relative to its counterpart in a battle, survivability and superiority may be maintained with the numerically inferior force. In order to improve the state of the art platforms, system integration must be considered from the outset. In the System Integration Branch of the Avionics Directorate, several programs are researching ways of improving weapon system capability by improving system cooperation. Within the Branch, programs are conducting research in integrated systems for the purposes of covert penetration, cooperative offensive and defensive electronic combat, and intelligent fusion of sensor input for the correlation and location of threat systems. Three programs within the Branch are involved with this arena of systems integration: Fused Avionics for Threat Identification and Location (FATIL), Evolutionary Electronic Combat Concepts (E2C2), and Quiet Knight. The objective of the FATIL program is to improve friendly aircraft survivability within a threat situation by fusing multiple data sources to enhance confidence in threat identification, to strengthen the accuracy of threat location, and to augment crew situation awareness. Under E2C2, the Covert Integrated Electronic Combat Controller (CINTEC2) has the objective of developing an expert system mediator between integrated offensive and defensive management systems by deconflicting contradictory actions during a mission. In Quiet Knight, completely integrated avionics suites will be flight tested to prove concepts for quicker and more accurate threat evasion in a performance-limited aircraft. In order to best pursue these areas, it was deemed necessary to perform a preliminary flight test program to collect electronic warfare data and valuable flight test experience. The Early Bird program was designed to meet these needs. In particular, the Early Bird program included the following objectives: to explore the correlation of threat location with a digital terrain database for display to the aircrew; to investigate real-time database updating based on 1553B bus data; to evaluate a scheme used for storing large digital terrain databases; and to evaluate display fidelity requirements for situational awareness.

1.2 Integrated Terrain Access and Retrieval System Overview. The Early Bird program involved the flight test of the Integrated Terrain Access and Retrieval System (ITARS). The ITARS is a flight ready system designed to handle the usage and storage of digital terrain elevation data (DTED) and digital feature analysis data (DFAD) developed by the Defense Mapping Agency (DMA). These two sets of data make available to any digital computer a model of the real world terrain and environment. In particular, the DTED and DFAD have stored in them the elevation of all areas of the world and the vegetation or cultural features associated with each area. Since this data is voluminous, it is necessary to have a database manager to process and store the data. The ITARS is capable of storing in memory four cells of DTED and DFAD simultaneously (where each cell is one degree in latitude by one degree in longitude). Data in excess of four cells may be stored on an optical disk system and loaded into the ITARS during a mission. The ITARS unit is capable of generating three simultaneous displays for crew usage: a heads up display (HUD), a multimode display, and a plan view display. The HUD image presents to the pilot a ridge line perspective view based on the stored elevation data. The plan view channel generates a sun angle shading view of the elevation data from an overhead vantage point. The

multimode channel is capable of generating all of the plan view modes plus perspective view modes for a heads down look at the area in front of the aircraft. For a more in depth description of the ITARS unit, see PAVE PILLAR In-House Technical Report, Volume I, Section 3.0.

1.3 Early Bird Program Overview. In Early Bird, the ITARS unit was integrated onto the Boeing Advanced Avionics Testbed (AAT). The Boeing AAT consisted of a full complement of avionics equipment. In Early Bird, the ITARS was used to display current aircraft position relative to the outside world, plus to display situational awareness data to the "crew" (ie, threat emitter locations and lethality rings). While these functions were tested, valuable electronic support measures (ESM) system data was being stored for future analysis. This data will be used to directly support the Quiet Knight, FATIL and E2C2 program efforts. Also, the flight testing experience gained will be used for the flying aspects of the Quiet Knight program.

2.0 Approach

2.1 Implementation Overview. The approach taken by the Early Bird Team involved flying on the Boeing AAT in a non-intrusive role as outlined in the Memorandum of Understanding (MOU). The MOU required that the Early Bird system monitor and extract existing messages from the AAT 1553B bus traffic. These messages would then be passed to the ITARS unit on a separate 1553B data bus after the messages were reformatted.

2.1.1 Early Bird Requirements. The Early Bird program allowed for the team to have equipment flying on the AAT which monitored and recorded real-time 1553B bus traffic during missions planned for Boeing's independent research and development (IRAD) programs. The Early Bird rack of equipment was operated as a stand-alone system. In this manner, the ITARS unit could be controlled without adding any message traffic to the Boeing 1553B bus system. In order to collect useful data, the Early Bird program required flying an aircraft with an ESM system against a typical threat environment (radar sights associated with surface to air missiles, SAMs, or simply emitters). The existing flight schedule involved a series of sorties over an electronic combat test range which was suitable for collecting the required data.

2.1.2 Flight Schedule. The Early Bird team and equipment was to fly piggyback on an existing flight schedule for Boeing's ongoing research (AAT Phase II Flight Sched-

**AAT Flight Test - Phase IIB**

| April 1990 | | | | May 1990 | |
|---|---|---|---|---|---|
| 1-7 | 8-14 | 15-21 | 22-28 | 29-5 | 6-12 |
| * local area | *Moses Lake | * Moses Lake | * Tyndall AFB | *New Orleans | * Mt Home |
| * check flight | * system test | * system test | *ANG tests | * ANG tests | * Full test |
| * 2 sorties | * 2 sorties | * 2 sorties | * 4 sorties | * 2 sorties | * 3 sorties |

**Figure 1**

ule). The initial flight schedule consisted of flights originating in Seattle, Washington (see flight schedule, figure 1). The Phase II schedule called for a total of 15 sorties at four locations: two at local area/Puget Sound, four at Moses Lake (airport area due east of Seattle), four at Tyndall AFB, Florida, two in New Orleans, Louisiana, that was used to simulate a threat site. Thus, these missions provided Early Bird with a single threat environment for debugging purposes prior to the Mt. Home flight. The Mt. Home flights were intended to provide a more realistic threat environment for evaluation purposes. There were to be multiple threats of varying types dur-

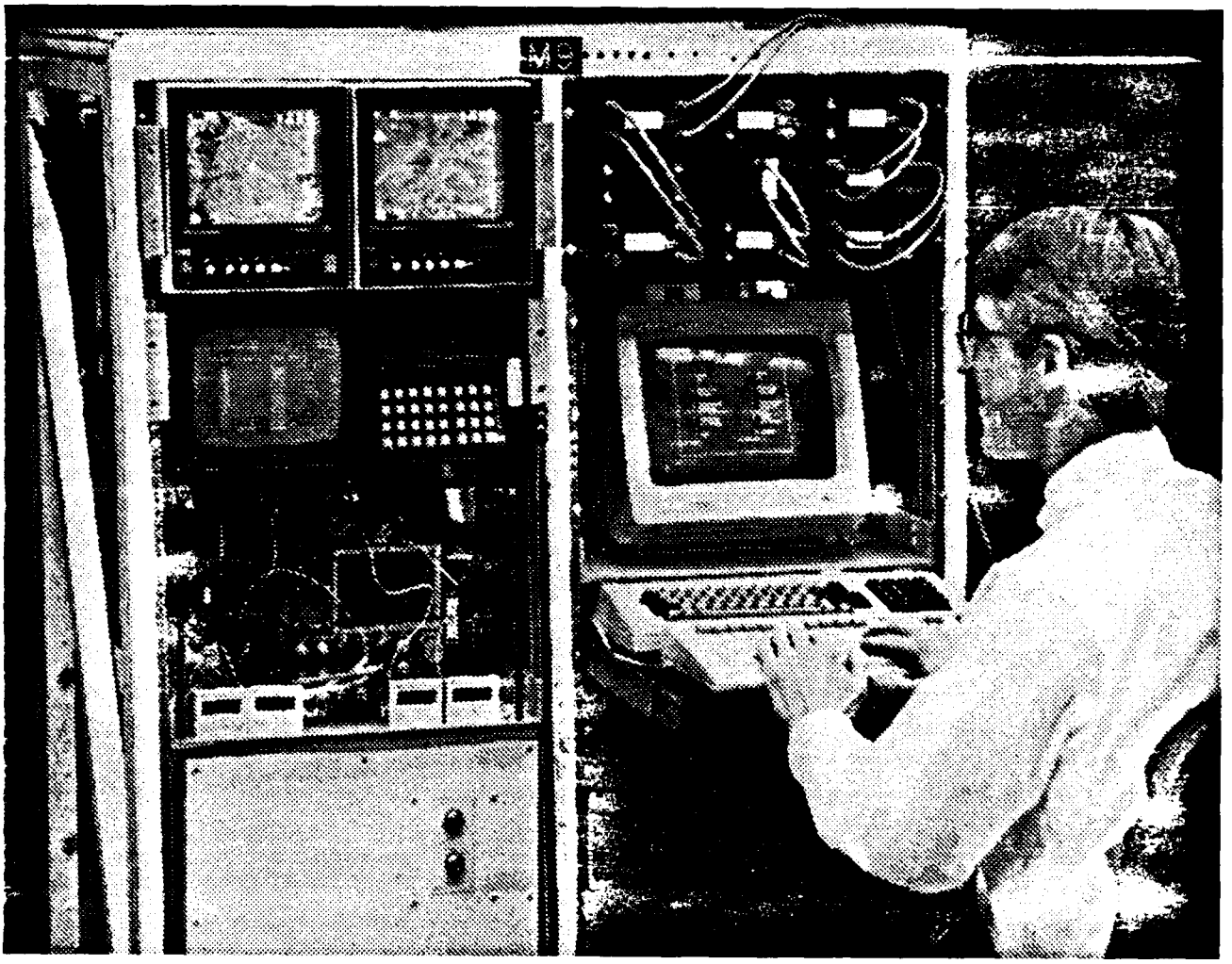


**Figure 2**

and three at Mountain Home Electronic Combat Range, in Idaho. The Early Bird team plan was to participate in all flights in the Seattle area prior to the three sorties at Mountain Home, which were the primary sorties of interest. The flights in the Seattle area involved flying low-level racetrack orbits in the mountain regions east of Seattle. One portion of the orbit included flying over the Moses Lake airport with a radar system ing each sortie. Thus, the hardware and software could be evaluated in an environment which is more characteristic of the real world threats. However, unexpected events at Boeing resulted in a reduced schedule and the elimination of the Mt. Home flights. The data collected from the Early Bird flights was not as comprehensive as was originally planned, but served to provide initial insight into several issues.

2.2 Hardware. The Early Bird flight test rack consisted of the Integrated Terrain Access and Retrieval System (ITARS) Flight Unit #3, the Early Bird MicroVAX with a VT-220 terminal, the Loral 1553B System Bus Analyzer, the ITARS Cooling Unit (ICU), and the data, video, and power distribution subsystems (see figure 2).

2.2.1 ITARS. The Naval Air Development Center (NADC) loaned the Air Force the ITARS Unit #3 for flight testing. Unit #3 has an enhanced internal air flow to promote better cooling of the ITARS components during flying operation. The additional ITARS Cooling Unit (ICU), required to maintain an operable internal temperature, is discussed in section 2.2.4. Unit #3 also has a redesigned card mounting system (required by a change in the cooling airflow in the unit) to reduce avionic failures due to shock and vibration. The major visible difference in Unit #3 is its use of roll/pitch stabilization on the perspective view of the multimode channel (the perspective view takes into account aircraft motion and maintains a horizon consistent with the outside world) and the addition of an optical disk drive unit (see figure 3). Two of the three ITARS video output channels were used for this project, the plan view channel and the multimode channel. ITARS was mounted in an environmentally stabilized plexiglass enclosure installed in the rack.

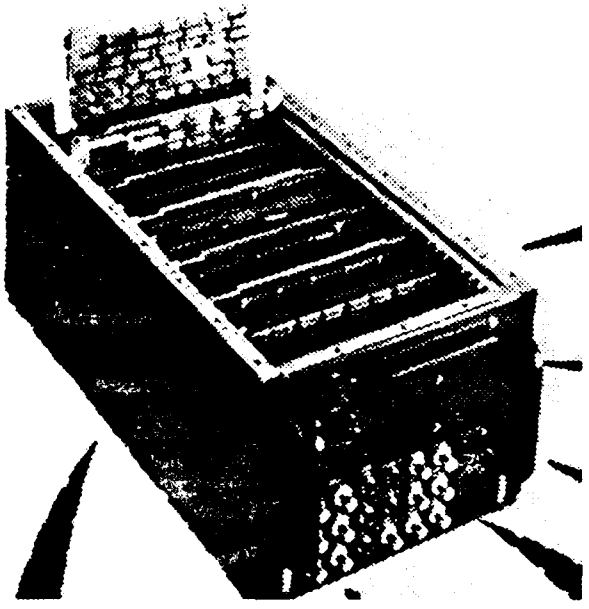
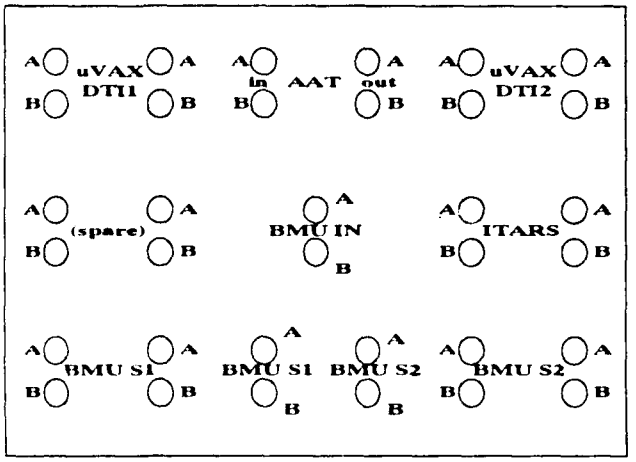


**Figure 3**

2.2.2 MicroVAX. The Early Bird MicroVAX (EBIRD) is a Digital Equipment Corporation (DEC) MicroVAX II housed in a ruggedized chassis with removable hard disk drives. The central processing unit (CPU), memory, Ethernet interface, and TK50 controller cards are standard DEC equipment. The SA-H169, a ruggedized chassis with fourteen quad-wide Q-bus slots and 200 watt power supply, and RQD-11 SCSI bus controller card (which controls the disk drive units of the MicroVAX) were manufactured by Sigma Sales, Inc. The 380 Mb Wren Runner hard disk drives were manufactured by Imprimis, and were housed in removable Sigma canisters with plug-in docking connectors. In addition to the hard disk drives, two tape drives, a DEC TK50 and an Exabyte 2000 8 mm, were installed in removable canisters for backup of the flight data and control software. Two Digital Technologies, Inc. (DTI) BIU-53Q BC/MRT cards were installed, one for 1553B data acquisition and the other for ITARS mode control. A DEC VT220 terminal was used as the operator's console.

2.2.3 Data Bus Monitor. A Loral SBA 100 1553B bus analyzer was used as a data monitor in the rack. The SBA 100 was used to monitor either the AAT bus or the ITARS bus by use of the 1553B patch panel. By monitoring the bus, the Early Bird team was capable of debugging software and message traffic errors in the system. The SBA 100 was also used to test ITARS threat displays during ground test when the AAT bus was inactive. This was accomplished by entering data into a 1553B message format recognizable to the ITARS and transmitting it from the SBA 100.

2.2.4 ITARS Cooling Unit. ITARS Cooling Unit (ICU) was designed especially for the flight test rack and the ITARS Unit #3. The ICU is a 4050 BTUH cooling unit operating in a closed loop environment. The ICU pulled heated air from the ITARS enclosure, cooled it, and returned the cooled air to the ITARS air intake. The cooling unit was able to maintain an average 93 degrees Fahrenheit at the ITARS exhaust during flight operations.




**Early Bird Patch Panel Layout**

**Figure 4**

2.2.5 Data Distribution. The data distribution subsystem was responsible for directing 1553B and Ethernet communications within the rack. The 1553B communications were routed to and from each 1553B component via coaxial cable through a central transformer-coupled patch panel (see figure 4). Transformer coupling allowed disconnection from the host AAT bus without interruption of signals. This coupling also allowed the SBA 100 bus analyzer to similarly be switched from the AAT bus to the ITARS bus without disruption of communications. The Ethernet communications from the MicroVAX to ITARS (used in downloading ITARS with software and DTED/DFAD databases) passed through a DESTA converter. The DESTA converted the thick-wire Ethernet output of the MicroVAX to the thin-wire input on ITARS.

2.2.6 Video Distribution. The video outputs from ITARS were connected to RS-170 to NTSC color video encoders to allow external video taping of the ITARS displays. From the encoders, outputs were also routed to two rack-mounted nine inch color video display terminals (VDT) for operator use. One VDT was dedicated to the ITARS plan view channel and the other to the multimode channel.

2.2.7 Power Distribution. Power was available on the AAT in the form of 400 Hz and three phase 60 Hz. From the aircraft, the power passed through circuit breakers to protect the flight test equipment from potential excessive current levels. The 400 Hz power was then routed to ITARS. The 60 Hz power was split into individual single phase lines, then made available for the equipment in the rack requiring 60 Hz power via standard three-prong plugs.

2.2.8 Flight Test Rack. The flight test rack was a dual bay flight-ready equipment rack provided to the Air Force by Boeing. The rack was designed for standard nineteen inch wide equipment and measured 42.5" high and 30 7/8" deep.

2.3 Software. The software developed for this project consisted of preflight, airborne, and post flight software.

2.3.1 Preflight Database Preparation Software. The Database Preparation Software changes were intended to address problems experienced during database preparation and to improve ITARS performance. These software changes covered the routines that process the linear and point features, in addition to the feature options files.

2.3.1.1 Database Preparation Software Issues. During the preparation of the databases for Early Bird, it was discovered that when a linear feature exited a block at one of its corners, the database preparation software generated an error and ended processing. To solve this problem the software logic was modified to test when a linear feature exits the corners of a block as well as the sides. Also, during this phase of the project another problem was encountered. The linear and point feature buffers were overflowing on the preparation software. During the investigation of the problem, it was discovered that DMA DFAD Level 1C contained too many linear and point features for the database preparation software to handle. To solve this problem, an increase in the size of the linear and point feature buffers was attempted. These buffer changes resulted in some other problems when the final database files were downloaded into ITARS. Specifically, the new files were larger than the space allocated for them on the ITARS system mass memory, which resulted in the deletion and/or corruption of pointer and application files during ITARS operation. In order to overcome the feature buffer problem, it was decided to filter the linear and point features by eliminating some of them from the DMA files. These features, and their Feature Identification Codes (FIC's), are found in Table 1. The feature option files contain the FIC for the features that will be processed by the database preparation software and combined into ITARS FIC groups.

| Feature Type | FIC | Description |
|---|---|---|
| Point | 0 | |
| Point | 99 | |
| Point | 120 | Refinery |
| Point | 182 | Smokestack |
| Point | 260 | Bridge |
| Point | 511 | Radio/TV Tower Type "A" |
| Point | 530 | Miscellaneous Towers |
| Point | 540 | Power Transmission Tower |
| Point | 801 | Tank |
| Point | 928 | Plaza/City Square |
| Linear | 750 | Maritime Features |
| Linear | 751 | Breakwater/Jetty |
| Linear | 752 | Wharf/Pier |
| Linear | 753 | Drydock |
| Linear | 925 | Dams/Weirs |
| Linear | 931 | Salt Water, Sea State |
| Linear | 940 | Fresh Water |
| Linear | 943 | Fresh Water, subject to ice |

**Table 1: Attributes Eliminated - First Modification**

2.3.1.2 ITARS Display Performance. Another issue that was solved by modifying the database preparation software was the ITARS display performance problem. This problem was detected during ground testing of the Early Bird system on the AAT in Seattle. It was discovered that for the gaming areas intended for the first flight there existed considerably more linear and point features than normally encountered in ITARS testing. This resulted in degraded performance of the ITARS displays; the displays showed mismatches between the terrain and the linear and point features associated with the terrain. This was characterized by bridges, roads, airports, and control towers moving back and forth over the elevation data as the aircraft moved.

| Feature Type | FIC | Description |
|---|---|---|
| Linear | 244 | Road Viaduct |
| Linear | 260 | Bridge |
| Linear | 261 | Suspension Bridge |
| Linear | 263 | Arch Bridge |
| Linear | 264 | Truss Bridge |
| Linear | 265 | Moveable Span Bridge |
| Linear | 267 | Deck/Bridge |
| Linear | 752 | Wharf/Pier |
| Point | 702 | Airport Control Tower |
| Point | 718 | Radar Antenna, Tower Mounted |
| Point | 790 | Military/Civil Associated Structure |
| Point | 811 | Bullet |

**Table 2: Features Eliminated - Second Set**

This problem is internal to ITARS and is a limitation on the processing power of one of the 1750A processors: the Point Feature Processor. In order to alleviate this problem, the feature option files were modified again to filter additional features. The additional FIC's eliminated are found in Table 2. The combined modifications generated a reduced set of linear and point features (see Table 3) for the database files that contributed to a lower workload and an increased performance for the Point Feature Processor.

2.3.2 Airborne Software. The Early Bird airborne software consisted of the ITARS start-up programmable read-only memory (PROM), download software, threat processing software, and the Early Bird MicroVAX software.

2.3.2.1 Start-up PROM. The Start-Up PROM software changes consisted of modifying the PROM so that the ITARS Ethernet address could be modified by issuing an ethernet message. This change was required in order to support the ITARS download from the MicroVAX, since the unit #3 address was incompatible with the Air Force software used. After this change was tested and verified, new PROMs were burned and installed on two of the ITARS 1750A processor boards.

2.3.2.2 Download Software. A new routine was created to download ITARS from a MicroVAX. In the Integrated

| Feature Type | FIC | Description |
|---|---|---|
| Linear | 245 | Causeway |
| Linear | 701 | Airport/Airbase |
| Linear | 706 | Airport Runways and Taxiways |
| Point | 701 | Airport/Airbase |
| Point | 710 | Airport/Airbase Electronic Navigation Aids |
| Point | 716 | Radar Antenna, Tower Mounted with Radome |
| Point | 760 | Maritime Navigation Aids |

**Table 3: Linear and Point Features - Reduced Set**

Test Bed facility, the ITARS unit is downloaded via personal computer. Since it was necessary to eliminate this hardware to save space in the flight test rack, routines were written to achieve a download using the flight test MicroVAX already in the rack.

2.3.2.3 Threat Processing. To support displaying threat symbols and lethality circles, the Point Feature Processor software files were modified. In particular, the process responsible for reading a message from system mass memory and interpreting it for validity was modified to allow feature types ranging from ITARS FIC 0 up to 63 (this is an increase over the eight feature types originally supported). This modification was needed to support 53 new feature types created for the threats, and to allow room in the event that it became necessary to add feature types which were not threats. Additionally, the 64 types are the maximum allowed by the 1553 message word. The process responsible for generating the ITARS display lists was modified to also allow feature types from ITARS FIC 0 up to 63. In addition, this process was modified to eliminate the processing of the overflow linear and point features. Finally, this process was heavily modified by adding the new feature types for the threats and by adding the software needed to generate the new threat symbols and lethality circles (the threats would be displayed on a map as a letter-digit pair surrounded by a circle, with the circle corresponding to the kill radius of the threat, and with the software translating the circle radius from nautical miles to pixels based on the display scale being used). The majority of the changes made pertained to the generation of a new display list for the Vector Generator.

2.3.2.4 Early Bird MicroVAX Software. The Early Bird MicroVAX software consisted of the bus controller card software, the monitor process, the controller process, and the user interface process.

2.3.2.4.1 DTI 1553B BC/MRT card software drivers. The 1553B bus monitor and control software consisted of the routines necessary to control the DTI BC/MRT cards (which are located in the MicroVAX and communicate between the MicroVAX and either the AAT or the ITARS over 1553B data busses). The DTI cards are programmed through a set of command and status registers (CSRs) initialized through a series of QIO calls configuring the card as either a 1553B bus monitor or controller. The 1553B software initialized he CSRs and the data input queue on the monitor card. Upon receipt of the specified incoming messages, the DTI card in the monitor mode delivers an interrupt to the host software. The card in the bus controller mode will transmit a message upon receipt of an interrupt from the host software.

2.3.2.4.2 Monitor Process. Upon receipt of an interrupt from the AAT bus monitor card, the AAT monitor software retrieved the incoming message, placing it into a working buffer. The monitor software processed the aircraft state vector and threat location messages from the AAT On Board Mission Manager (OBMM). The monitor software then decoded the messages, placing the pertinent information into a shared data area called a data commons. A more detailed analysis of the Monitor Process can be found in the Appendix.

2.3.2.4.3 Controller Process. The ITARS control software retrieved the data from the commons and formatted them as ITARS messages. The control software passed the messages to the 1553B controller which transmitted them onto the ITARS bus. A more detailed analysis of the Controller Process can be found in the Appendix.

2.3.2.4.4 User Interface Process. The operator interface software provided the operator with a means of loading DMA databases into ITARS and controlling the video output channels. ITARS control was accomplished through a set of hierarchical menus. Features which could be controlled by the operator included: gaming area selection and database download, plan view channel functions, and multimode channel functions. The plan view channel functions allowed the user to select display mode, map scale, map orientation, and map centering. The multimode channel functions include all of the functions of the plan view channel, plus the ability to select the amount of features being displayed and the ability to view a perspective display based on aircraft attitude. The operator interface software translated the operator commands into ITARS mode control messages, placing them into the data commons. A more detailed analysis of the User Interface Process can be found in the Appendix.

2.3.3 Post Flight Data Conversion Software. The nature of the program required that all incoming and outgoing messages be logged into archival files for later analysis. To minimize the time and space used for storage, the messages were written to disk using a VAX-specific binary format. The post flight software converted the binary files into human-readable ASCII files for quick analysis.

3.0 Lessons Learned. The following paragraphs identify the advantages, limitations, support tools, and possible applications of the systems utilized during the Early Bird project.

3.1 ITARS Database Issues. It was clear from the observations obtained during the systems integration and flight tests that future moving map systems need more and better capabilities than the ones provided by ITARS. Future moving maps and cartographic data management systems should incorporate the following features: an improved approach for the access, retrieval, and storage of larger cartographic databases (greater than the 100 X 100 nautical miles (nmi) supported by ITARS); larger display scale ranges (up to 200 nmi, if possible) to support the display of threats with detection and lethality ranges greater than 40 nmi; the ability to draw detection, lethality, and terrain masking data for threats located outside the display area; improved correlation between threat detection and lethality radii and the circles drawn to represent them (current ITARS display screens only have 256 x 256 pixels resolution which affects the position of circles and threats on the display); ability to download Electronic Order of Battle (EOB) data and threat parameters data; and finally the use of a standard set of symbols and encoding for the different threats and threat modes.

3.1.1 Database Paging. Currently, ITARS stores DTED in up to four cells of memory (where each cell is one degree in latitude by one degree in longitude) in BTC (Block Truncated Code) format. Beyond this, data can be stored in unlimited quantities on an optical disk in files of up to four cells each. To handle these volumes of data requires that all ITARS functions halt until download of a new database is complete. While the blocks of data are large and not loaded often, a given download takes much time. Also, at this time ITARS has no built-in capability to automatically load a new database as the edge of the current database is approached (that is, the download is commanded externally to ITARS). One method of improving retrieval would be to store and retrieve data in smaller-size blocks (breaking the one degree by one degree cells into one-eighth of a degree by one-eighth of a degree, for example). This would make individual downloads shorter and other functions

| Type | Description | Color Code | ASCII Code |
|---|---|---|---|
| 1 | Towers/DVOF | 47 (dark brown) | 8 |
| 2 | Water Tanks | 42 (white) | 77 |
| 3 | Power Tower | 43 (red) | 9 |
| 4 | Military Structure | 44 (yellow) | 7 |
| 5 | Radar | 43 (red) | 6 |
| 6 | Waypoint Symbols | 43 (red) | 2 |
| 7 | Control Tower | 41 (blue) | 2A |
| 8 | Airport | 43 (red) | 1 |

**Table 4: Original Point Feature Attributes**

would not be interrupted. Another improvement would be to allow ITARS or the database manager to load new segments of data automatically before they are needed. Note that this process should not interrupt any ITARS functions (displays, terrain following, profile generation, etc.). While ITARS is downloading the new database from the optical disk, the displays and output data are frozen until the download is completed. This is an unacceptable situation for a moving map or a cartographic database manager that is providing terrain and feature data to time critical algorithms like Terrain Following, Terrain Avoidance, Threat Avoidance, Route Replanners, and Threat Intervisibility. Data should be stored in an uncompacted format, if possible, to alleviate decompaction errors, and should be stored in RAM modules or evolving technologies other than disk/tape media for faster direct memory access (DMA) transfer of data to mass memory usage (in ITARS). To correct this shortfall, an appropriate strategy is needed where database creation and storage, paging techniques, and look-ahead are considered when choosing alternatives for the management of big cartographic databases.

3.1.2 EOB Implementation. The original ITARS software did not provide for an EOB. The EOB was implemented for Early Bird using the waypoint list table, which was originally limited to eight feature types with each type having an associated color code and ASCII code (shown in Table 4). The color code represents the color of the feature type, while the ASCII code is the internal representation of the feature. These codes were arbitrarily assigned by Hughes and are located in the Vector Generator PROM. The original eight feature types were then expanded to include threat types, of which there were 53, bringing the total number of types to 61. Currently, the original eight feature symbols are burned into the Vector Generator PROM. The new threat symbols and kill ranges are hard coded into the Point Feature Processor software. In a fielded system, it would be desirable to store the symbol set in PROM with the detection and kill range information downloaded in software. The initial EOB is stored as part of a waypoint list (which is the total of 63 waypoints and emitters allowed). To improve upon the current situation, it would be desirable to have unlimited EOB database size, as well as unlimited numbers of each EOB type, and to keep a database of all known emitter characteristics. In addition, the initial EOB data should be kept separate from, and should not be limited by, the waypoint list. Also, the database management system needs the capability to transfer detected EOB elements to the ground crew after a flight, as well as the capability to display threat intervisibility

patterns regardless of map display scale. The detection and kill ranges of threats should not be hard coded but should be kept in a separate classified file. In addition, a mission file that could be downloaded at the start of a mission with separate databases including waypoints, known threats, and targets should also be created.

3.2 Software Development. The software development area for real-time, multi-processor embedded systems was an area where the ITARS system lacked the necessary tools. The only tool available to help in the development of the ITARS software was the ITARS User's Console. The ITARS User's Console is a data probe that connects to a bus on top of the ITARS 1750A CPU cards and allows the user to examine the CPU registers, local memory, and the system mass memory through software running on a PC-AT. This tool proved useful in a certain number of cases, but it lacked the necessary capabilities to perform single-stepping through the software, to set break points, or examine variables by their logical name. Using the ITARS software development case as a reference, it was determined that a better software development environment is needed for developing, testing, debugging, and modifying real-time, multi-processor embedded system software. For this kind of system, the software developer needs a set of "user friendly" tools which support debugging the software while it is running in the targeted hardware and interacting with more than one processor. This set of tools should allow single stepping through the software, setting of break points, enabling wait-on-interrupts for selected events, accessing of program variables by logical names or physical addresses, and accessing of system local memory, global memory and registers. These tools should be targeted for multi-processor systems, where software running in multiple processors communicate with each other through the use of messages, flags, and interrupts, as well as Ada rendezvous. Also, these tools should be able to show the states and actions of the multiple processors through the use of windows.

3.3 Uses of Moving Maps and Database Manager Systems. Future moving map systems should be considered as a possible alternative for accomplishing navigation updates. The out-the-window terrain features can be used as reference points to be identified in the moving map and used as update points for the navigation system. These additional capabilities will make future moving map systems more flexible and capable of improving the pilot situational awareness for low level navigation and threat avoidance, especially if the moving maps are going to be used as the fusing element for threat data.

4.0 Conclusions. The Early Bird project resulted in an excellent opportunity for hands-on experience in the areas of software modification for real-time embedded systems, moving map evaluation, threat information overlay, data recording, and avionic systems integration. By working on these different areas, the Early Bird team obtained a clear understanding of the ITARS capabilities and limitations, the support tools necessary to accomplish software modifications for embedded systems, and the practices necessary to accomplish a successful avionics integration and flight testing. The lessons learned from this project will help in the development of more flexible and capable avionic systems.

# Appendix: Detailed Software Analysis

**Barbara L Eldridge**

Appendix: Detailed Software Analysis

1.0 Overview. The software development involved generating interfaces which could pull data off the AAT bus, send the data to ITARS and change the moding on the ITARS displays. The process that captured data off the AAT bus was labelled the Monitor Process, while the process that sent the updated aircraft position and mode changes to ITARS was the Bus Controller Process. The process that allowed the user to download new databases, change modes on ITARS, send waypoints and threats to ITARS, and request ITARS system health was labelled the User Interface (UI) Process.

2.0 Common Area Interfaces. The three processes of the Early Bird system were linked together with five separate common areas. (see figure 1). A Common Area is a disk file that resides in physical memory and contains data that can be made available to a process or processes for manipulation and execution. The five common area interfaces consisted of three Global Common Sections (GCS) and two local common areas. A Global Common Section is a copy of data in memory that allows all the processes connected to it, to access the same copy of data. Because the data is global, all processes who use it have to exercise resource allocation principles. The three processes shared global sections of memory, which contained a layout of messages such as the aircraft latitude, longitude, and altitude along with emitter and waypoint latitude, longitude, and altitude. These pieces of data were required by ITARS to update its aircraft position, and to display the threat and waypoint data. The process configuration, including data flow and major interfaces, is shown in figure 2. The three Global Common Sections (GCS) were shared and maintained by each process using it, and all data traffic off either the AAT 1553B bus or the ITARS 1553B bus was logged in a data

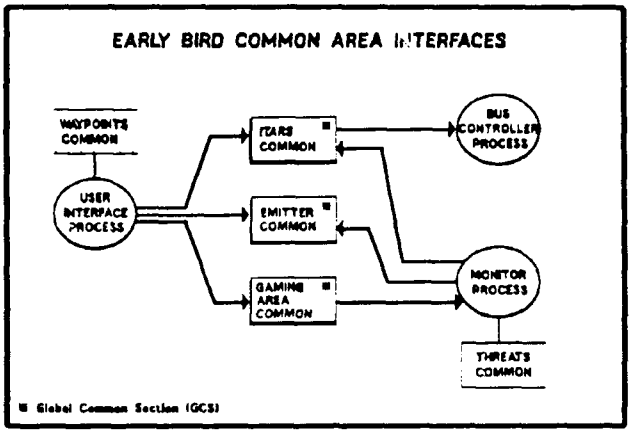



**Figure 1**

file for post-processing analysis. The ITARS Download program which downloads the data base to the ITARS Unit is discussed in the main text.

2.1 ITARS GCS. The ITARS GCS contained the four messages needed to update the ITARS unit. Access to the ITARS GCS was controlled through a flag buffer.

2.1.1 Data Words. There were four message buffers in the ITARS GCS: STATEVEC,

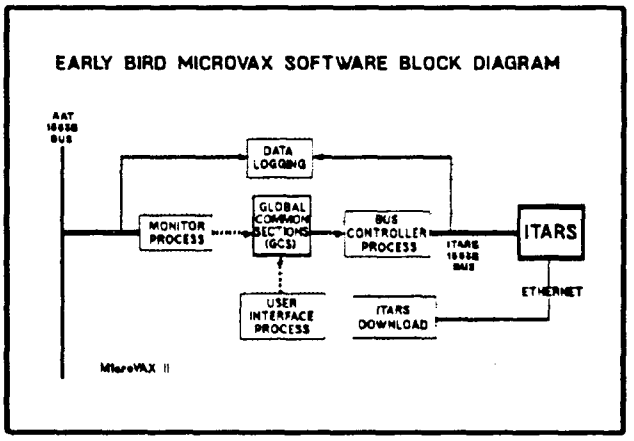



**Figure 2**

MODECTL, FEATURE, and ITARSSTATUS. STATEVEC was a 32-word synchronous input message buffer which contained all the navigation and aircraft state vector parameters necessary for ITARS operation. This buffer used the data captured by the Monitor Process which

was placed in the STATEVEC buffer in the ITARS GCS. This data was also logged in a data file. The Bus Controller Process took the data from the STATEVEC buffer in the ITARS GCS and sent it to ITARS. MODECTL was a 12-word asynchronous input message buffer which allowed the user to change the ITARS mode of operation. This message data was generated and placed in the ITARS GCS by the UI Process when the user changed the ITARS mode. The Bus Controller Process took the data from the MODECTL buffer in the ITARS GCS and sent it to ITARS. The 26-word asynchronous input message FEATURE provided the control structure to accomplish the following ITARS point operations on waypoints and designated points: add a point, delete a point, sequence the waypoints, or delete all waypoints and designated points. This buffer contained threat data from the Monitor Process, or it contained waypoint data from the UI Process. The Bus Controller Process took the data from the FEATURE buffer in the ITARS GCS and sent it to ITARS. A 15-word asynchronous output message buffer called ITARSSTATUS provided the status information to the user upon demand. Initiated by the user, the UI Process requested that the Bus Controller Process send a 1553 message to ITARS requesting that ITARS return the status of its processors. The Bus Controller Process then waited on the data from ITARS and, once received, placed the data in the ITARSSTATUS buffer. The UI Process decoded the data and displayed the information for the user. The ITARS GCS also contained an integer word ASYNC_MSG, which was set by the UI Process to tell the Bus Controller Process whether the Synchronous message being sent to ITARS was the 12-word MODECTL or the 26-word FEATURE buffer. This was necessary so that the Bus Controller Process knew which of the two asynchronous bus lists (the 12-wo%w or 26-word one) to send out on the 1553B data bus. The bus list is the control structure which contains the message to be sent out on the 1553B data bus to ITARS. The bus list contains the instruction type (master-to-remote or

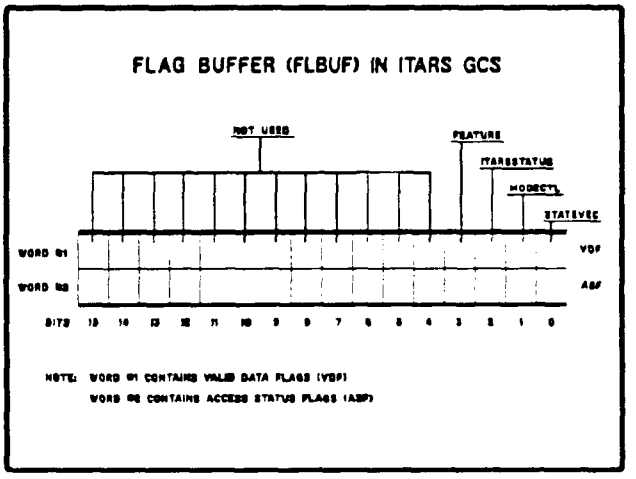



**Figure 3**

remote-to-master), the remote terminal, subaddress, and word count information, the data buffer to be sent, and finally a halt command. The MODECTL, FEATURE, and STATEVEC buffers were defined as Master-to-Remote instruction types since the messages were sent from the Bus Controller Process (master) to ITARS (a remote terminal). The ITARSSTATUS buffer was a Remote-to-Master instruction type since the message came from ITARS to the Bus Controller Process. Each buffer had its own bus list, since the buffers were of different sizes and were targeted to different subaddresses.

2.1.2 Flag Buffer. The flag buffer, FLBUF, was designated as two 16-bit words that were used to implement resource allocation and data validity checks/verifications for the four buffers in the ITARS GCS (see figure 3). In the two words of FLBUF, the first four bits were each designated to control one of the ITARS GCS buffers. Bit zero was used to control the STATEVEC buffer, bit one the MODECTL buffer, bit two the ITARSSTATUS buffer, and bit three the FEATURE buffer. The first word in the Flag Buffer was designated the Valid Data

Flag word. The bits in this word were set if a data buffer contained valid data and cleared otherwise. For example, if the INS latitude word within the STATEVEC buffer had new data placed in it by the Monitor Process, the bit corresponding to the STATEVEC buffer in the Valid Data Word of the Flag Buffer would be set (bit zero of word one). The second word in the Flag Buffer was called the Access Status Flag. This word was used to force mutual exclusion. If one of the processes was reading or writing into a buffer, then that bit of the Access Status Flag would be set. For example, if the Monitor Process was placing new data into the INS latitude buffer, the Monitor Process would set the bit corresponding to the STATEVEC buffer in the Access Status word (bit zero of word two), place the data in the buffer, and then clear the Access Status Flag for the STATEVEC buffer. Since the processes were required to check the Access Status Flag and set it before the process could read or write into a data buffer, it was impossible for multiple processes to access the same buffer simultaneously.

2.2 GAMING AREA Global Common Section. The GAMING AREA GCS contained three words. The first integer word contained the database number of the two-degree latitude by two-degree longitude gaming area being utilized. The other two words contained floating point numbers representing the latitude and longitude of the lower left corner of the database. The values for the block number, the latitude and longitude were determined prior to sorties, then loaded into a datafile. During a mission the software could command the optical disk to download a new block area to ITARS, based upon aircraft location. This GCS was accessed by the UI Process when a user wished to change the database area being displayed by the ITARS unit. When the user chose a new database area, the latitude and longitude were loaded into the GAMING AREA GCS. The Monitor Process used this GCS when it received new emitters off the AAT 1553 databus. The Monitor Process checked to make sure that the new emitters were within the latitude/longitude constraints of the database.

2.3 EMITTER Global Common Section. The EMITTER GCS contained information used when sending new point operations to ITARS. This GCS was designed as a database which contained the latitude, longitude, altitude, range error, emitter file number, sent-to-ITARS flag, initialization flag, and point feature type of an emitter. Functions performed on the database included updating an emitter's latitude and longitude, creating a new emitter, or deleting an old emitter. The database held integer values for the emitter file number provided by the OBMM message when a new emitter was defined. This emitter file number (0-32 on the AAT bus) was mapped into the point feature numbers available to ITARS (32-63) and taken from ITARS ICD. This limited the number of point operations to 32. The sent flag was used by the database to signal when an emitter had been sent to ITARS to display. This logical variable represented whether or not the emitter had been sent out on the 1553B data bus to ITARS. This GCS was used by the Monitor Process when it received new emitters off the AAT bus and by the UI Process when the user chose to enter emitters.

2.4 WAYPOINTS Common. The WAYPOINTS Common was designed as a database to maintain flight profile waypoints. The waypoint database was set up as a linked list where each entry in the database contained a pointer to the next waypoint in the list. Fields maintained for each waypoint in this database included the latitude, longitude, and altitude of the waypoint, the waypoint number as defined by the user, a type field for the point feature type, a valid flag, and a sent field for the

waypoint. In this database, the type field referred to the point feature type used by ITARS. The point feature types available to ITARS, for this flight test were labelled 0-31 for waypoints and 32-63 for emitters. The UI Process handled the valid flag which was used to mark out-of-sequence waypoints. When the user entered a waypoint, it may not have been in the sequence of mission waypoints. In this instance, the waypoint was defined but need not be displayed on ITARS, therefore the valid flag was set false. The last field of the database was the sent field, which designated whether or not the waypoint had been sent on the 1553B data bus to ITARS. There were two other words in the WAYPOINTS Common that were not part of the database: the head variable containing the pointer to the first waypoint in the linked list and the waypt_flag variable used for early debugging. Since only the UI Process was able to enter waypoints for the mission, the Waypoints Common was used only by that process.

2.5 THREATS Common. The THREATS Common was designed as a database, much

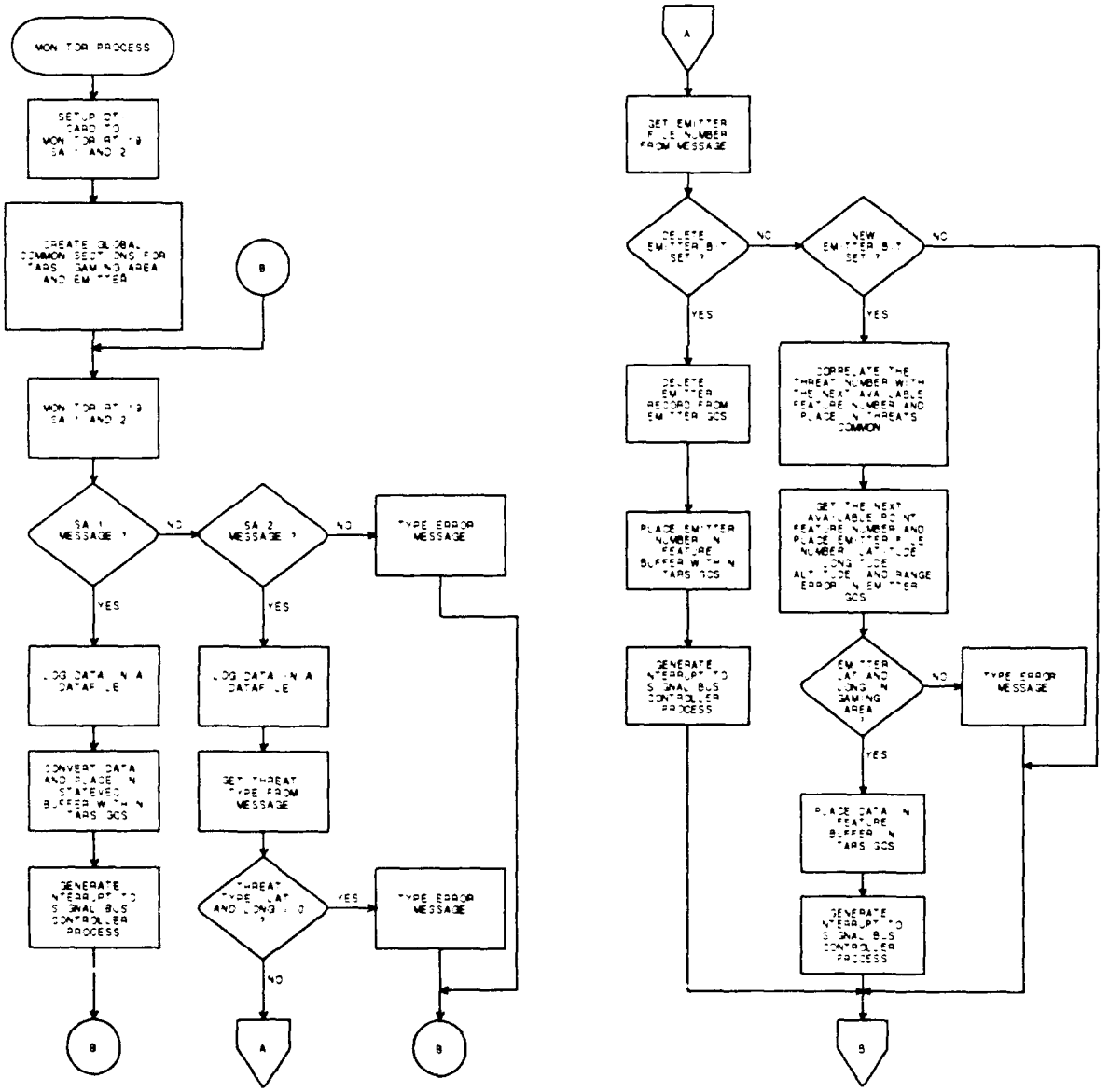



**Figure 4: Monitor Process Flowchart**

like the WAYPOINTS Common, to correlate the threat types received off the AAT bus to a feature type available for ITARS to use. This database maintained records which contained the threat number and feature number for a given threat defined by the OBMM. The threat type was in the range 0-256 while the feature type was limited to 11-60. This common area was internal to the Monitor Process when it received the threats off the AAT bus.

3.0 Software Processes. Three processes were developed to handle all the data capturing, reformatting and transmitting on the Early Bird project. These three software processes were the Monitor, Bus Controller, and User Interface Processes.

3.1 Monitor Process. The Monitor Process captured the Defense Management System (DMS) to On-Board Mission Manager (OBMM) message called "OBMM NAV DATA FILE" by monitoring Remote Terminal (RT) 19, Subaddress (SA) 1. This 32-word message was transmitted on the AAT bus at a 16-Hz rate, and contained the following data: INS latitude, INS longitude, true heading, roll, pitch, inertial and pressure altitudes, and ground speed. Another DMS to OBMM message containing the "OBMM EMITTER DATA FILE" was captured by monitoring RT 19, SA 2. This 32-word message was sent asynchronously and contained an emitter file number, threat type, and the latitude and longitude of the emitter. When new data for SA 1 or 2 was received on the AAT bus, the Monitor Process was signalled by an interrupt from the Bus Interface Unit (BIU) hardware. Once the interrupt was cleared and reset, the data received off the AAT bus was logged by the Monitor Process in a data file for post-flight analysis. The flowchart for the Monitor Process is shown in figure 4.

3.1.1 Navigation Data. When the data was from SA 1 (navigation data), it was converted to the appropriate units as utilized by ITARS. For example, the INS latitude was converted from a 32-bit word with a parameter range of plus or minus pi radians to a 32-bit word in units of degrees. Once the data was converted, the Monitor Process waited for access to the STATEVEC buffer. Finally, the Monitor Process set the Valid Data Flag, reset the Access Status Flag, placed the data into the STATEVEC buffer within the ITARS GCS, and generated an interrupt to signal the Bus Controller Process.

3.1.2 Emitter Data. The Emitter data captured by the Monitor Process from SA 2 (emitter data) contained either an add emitter or a delete emitter command. The emitters were stored and maintained in a local database called THREATS which was local to the Monitor Process. It allowed the Monitor Process to keep track of the Emitters as they were defined or deleted by the OBMM Emitter Data File.

3.1.2.1 Add Emitter. When the captured data was from SA 2 (emitter data) and the threat type, latitude and longitude fields in the message were defined (not zeros), then the emitter file number was extracted from the message. If the new emitter bit was set and the threat number was not already in a THREATS database, then the next available feature number from the database was obtained allowing the feature and threat numbers to be added to the THREATS database. The EMITTER GCS database was then searched for the emitter file number. As long as the emitter file number was not in the database, the point feature number, emitter file number, latitude, longitude, altitude and range error values were added to the EMITTER GCS database. After all the unit conversions were completed (from pi radians to degrees), the Monitor process again checked the Valid Data Flag, and the Access Status Flag before placing the data into the FEATURE buffer. The

process then set the Valid Data Flag, cleared the Access Status Flag, and generated an interrupt to signal the Bus Controller Process.

3.1.2.2 Delete Emitter. When the data captured by the Monitor Process was for SA 2, and it was not a new emitter, the process checked to see if the "delete emitter" bit was set. If the "delete emitter" bit was set, the data for that emitter was deleted (cleared) from the EMITTER GCS database. The process then checked the Valid Data Flag and Access Status Flag for the FEATURE buffer. After the Access Status Flag was set, the point feature number assigned to that particular emitter was placed in the FEATURE buffer. The FEATURE buffer was set up to send a delete emitter command, the Valid Data Flag was set, the Access Status Flag cleared, and an interrupt was generated to signal the Bus Controller Process.

3.2 Bus Controller Process. The job of the Bus Controller Process was to send the asynchronous and synchronous data to the ITARS unit over the 1553B data bus. Synchronous data is data that is delivered at a periodic rate. For example, in this project the aircraft state vectors were delivered from the OBMM at a 32-Hz rate. The three types of asynchronous data used were a mode control message that was delivered only when the user changed the moding of ITARS, the point feature/mission data that

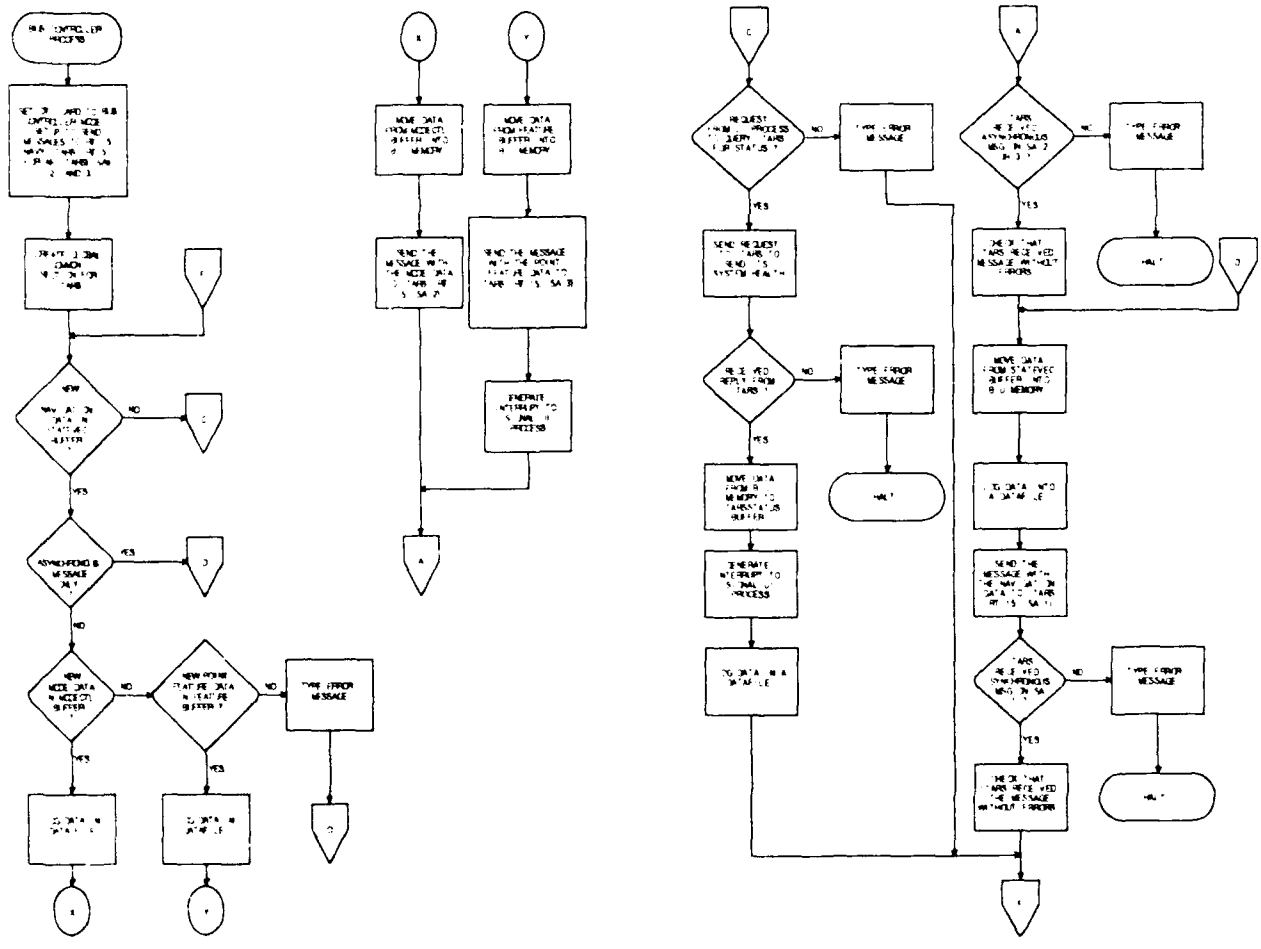


**Figure 5: Bus Controller Process**

was delivered when new emitters were sent from the OBMM across the 1553B bus or when the user entered waypoints or emitters, and the ITARS status data that was returned from ITARS when the user requested it. The Bus Controller Process continuously looped waiting on interrupts signalling new synchronous or asynchronous data off the AAT or ITARS 1553B data buses. The synchronous messages were trapped off the AAT bus by the Monitor Process, placed in the ITARS GCS and an interrupt was generated to signal the Bus Controller Process to begin execution. The asynchronous messages could come from either the AAT or ITARS 1553B data buses. The asynchronous messages for the point features and mode control messages were trapped off the AAT bus by the Monitor Process, placed in the ITARS GCS and an interrupt was generated for the Bus Controller Process. The asynchronous message off the ITARS 1553B bus was trapped by the Bus Controller Process and immediately processed. The priority was given to the asynchronous messages because asynchronous data comes only once and needs to be processed immediately. Missing one of the synchronous messages would not cause a large data inconsistency problem to occur since the synchronous message was delivered at a 16-Hz rate. Missing an emitter definition which only comes once would be impossible to recover. To perform these functions the Bus Controller Process mapped into the EMITTER GCS. For a flowchart of the Bus Controller Process, see figure 5.

3.2.1 Asynchronous Data.

3.2.1.1 Asynchronous Data from Monitor Process. The Bus Controller Process was triggered upon the receipt of new synchronous data since it was periodic. When the Monitor Process generated a synchronous interrupt, the Bus Controller Process cleared it and checked to see if another interrupt was set for asynchronous data. The asynchronous interrupt would be generated by the UI Process when there was a new mode control message (MODECTL) buffer to be sent to ITARS, or by the UI or Monitor Processes when there was new point feature data (FEATURE buffer) to be sent to ITARS.

3.2.1.1.1 New MODECTL Buffer. If the UI Process generated an interrupt for a mode control asynchronous message, the Bus Controller Process began by clearing the interrupt, checking the Valid Data Flag, waiting on the Access Status Flag for the MODECTL buffer to be cleared, setting the Access Status Flag, and moving the data from the MODECTL buffer to the BIU memory in the specified buslist. The Bus Controller process then cleared the Valid Data Flag and Access Status Flag for the MODECTL buffer. The buffer was sent out on the 1553B bus to ITARS and logged in a

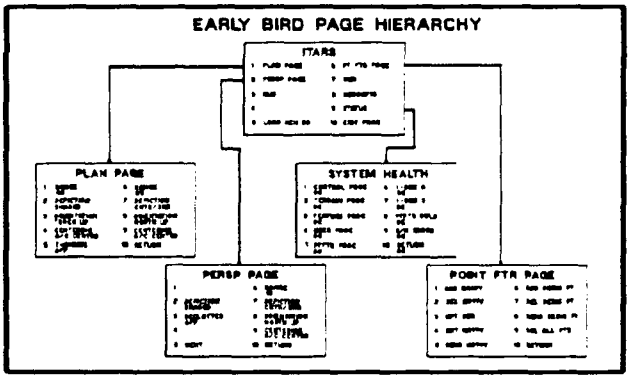



**Figure 6**

datafile. Finally, the Bus Controller Process then waited on a reply from the ITARS (RT 5 SA 3) stating whether or not it received the buffer correctly.

3.2.1.1.2 New FEATURE Buffer. If the interrupt was generated by the UI or Monitor Processes for point feature asynchronous data, the interrupt was cleared, the data was logged in a datafile, and the process waited on a set Valid Data Flag and a cleared Access Status Flag. The process

then set the Access Status Flag, moved the data from the FEATURE buffer to the BIU memory, and finally cleared the Valid Data and Access Status Flags. The buffer in BIU memory was then sent out on the 1553B bus to the ITARS unit. The process also generated an interrupt at this point to signal the UI process that the FEATURE buffer had been sent to ITARS and was available for processing again. Since waypoints can only be sent 3 at a time (each waypoint required 8 words in a buffer; 3 multiplied by 8 is 24 out of a 26-word buffer), more than one feature buffer must be sent to ITARS when handling four or more waypoints at a time. This was accomplished by using an interrupt to signal the UI Process, since the UI Process was set to wait on the interrupt in order to place more waypoints in the FEATURE buffer. After the FEATURE buffer asynchronous message, had been sent to ITARS, the Bus Controller Process waited on a reply from ITARS (RT 5, SA 3) stating whether or not it received the buffer correctly.

3.2.1.2 Asynchronous Data from User Interface Process. If the original interrupt was not generated by the Monitor Process, then it was generated by the UI Process requesting the system health of ITARS. In this case, the interrupt was cleared, a 1553 message was sent to ITARS commanding ITARS to send its system health, while the Bus Controller Process waited on the reply. The BIU was set up to generate an interrupt when the ITARS status message was received off RT 5 SA 5. When the reply was received, the Access Status Flag for the ITARSSTATUS buffer was monitored until it was able to be set, the data was placed in the ITARSSTATUS buffer. after which the Valid Data Flag was set and the Access Status Flag was cleared. An interrupt was then generated to signal the UI Process to begin processing the data in the

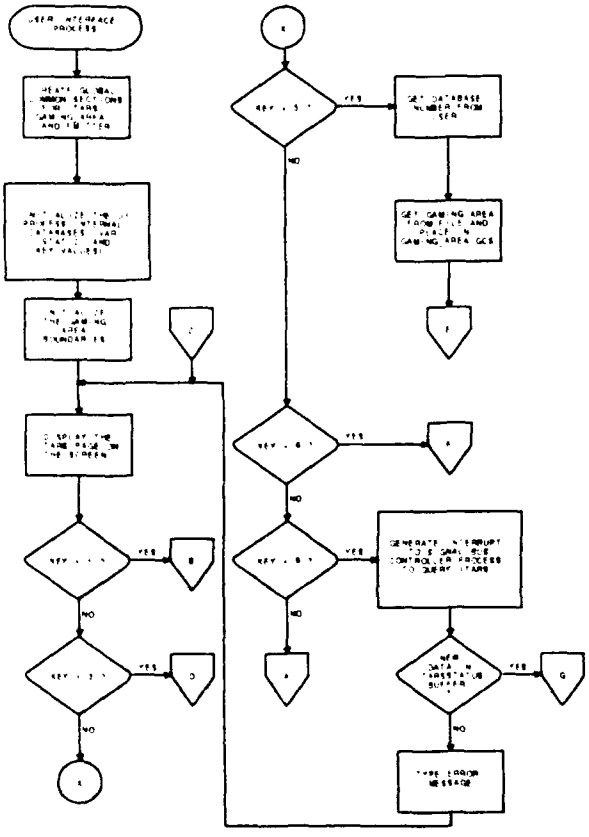


**Figure 7(a): User Interface (UI) Process**

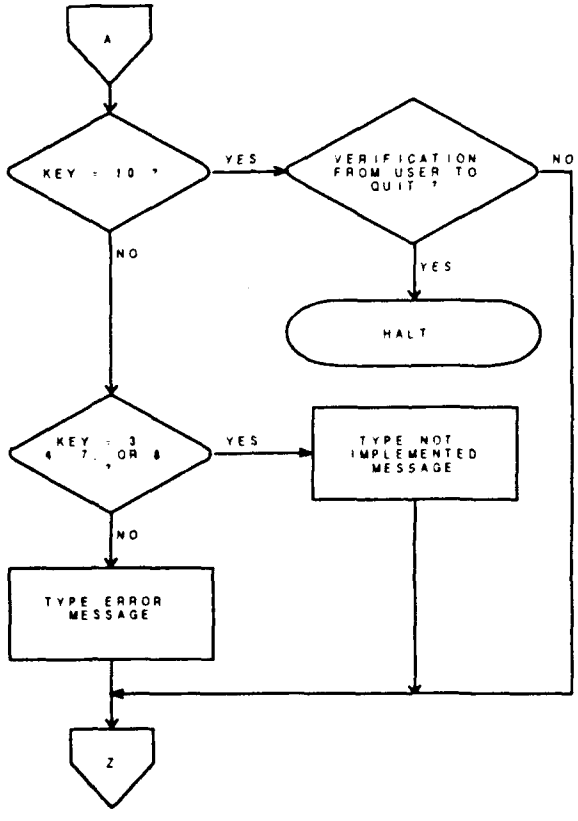



**Figure 7(b): UI Process**

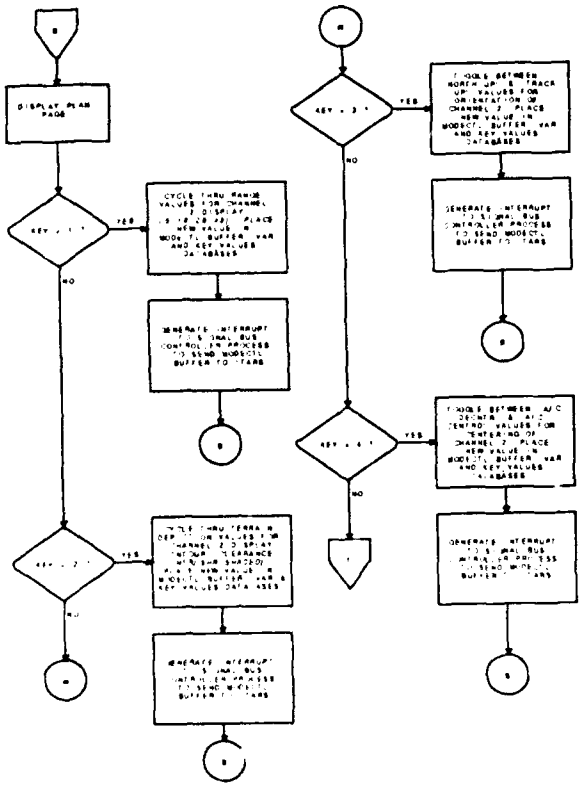


Figure 7(c)

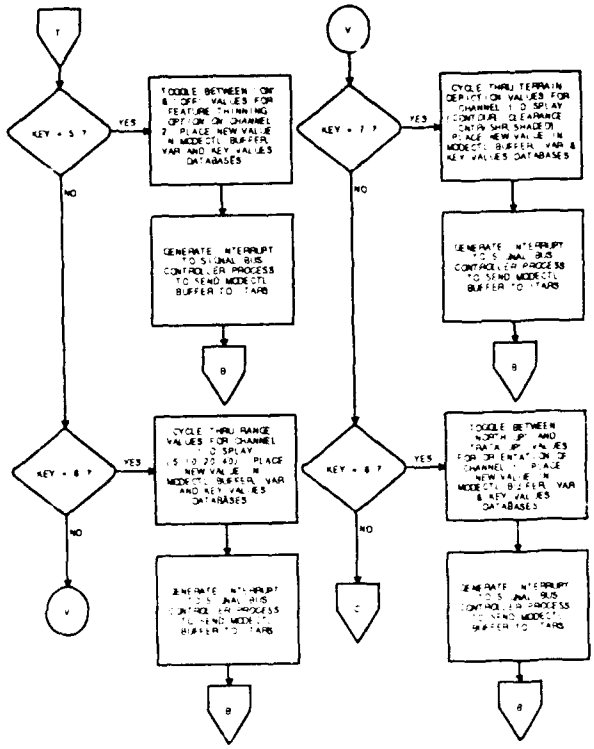


Figure 7(d)

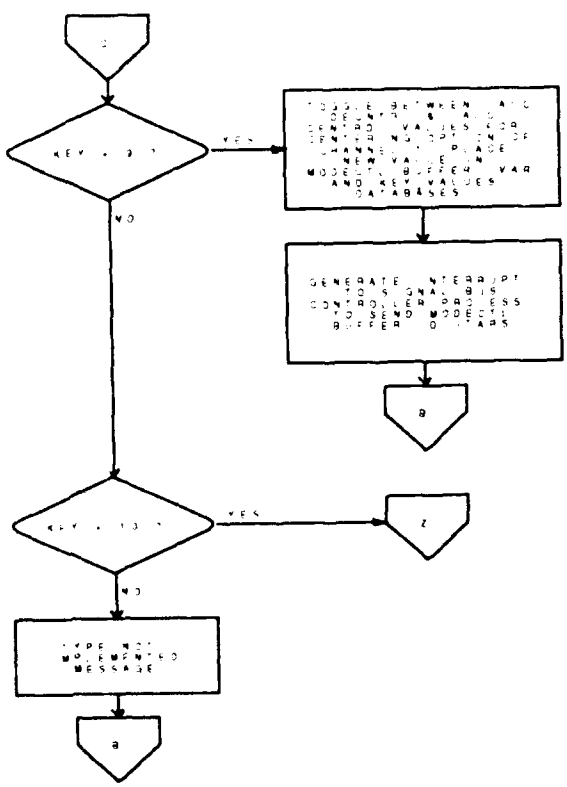


Figure 7(e)

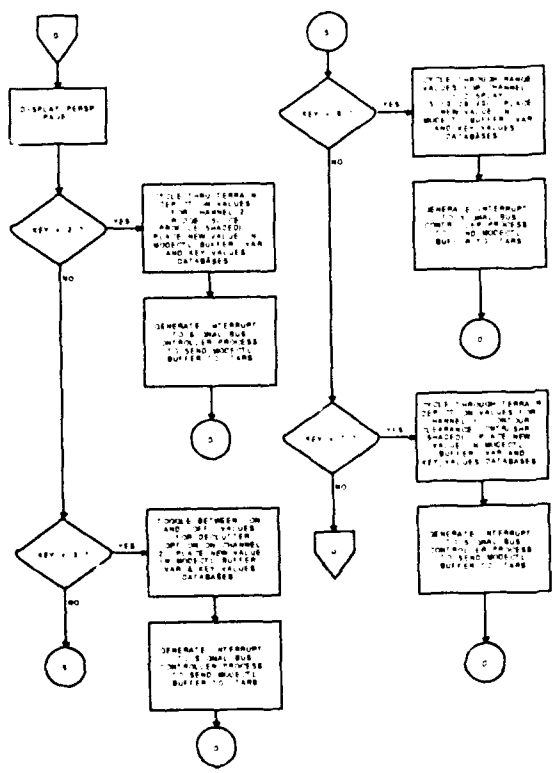


Figure 7(f)

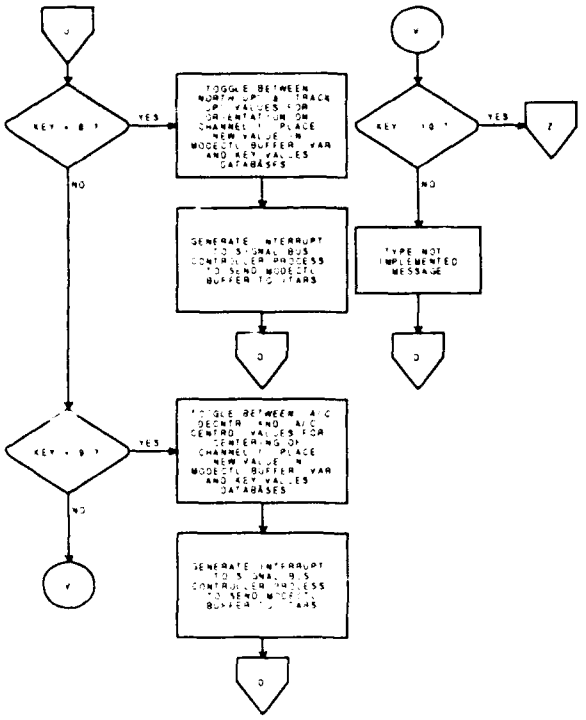


Figure 7(g)

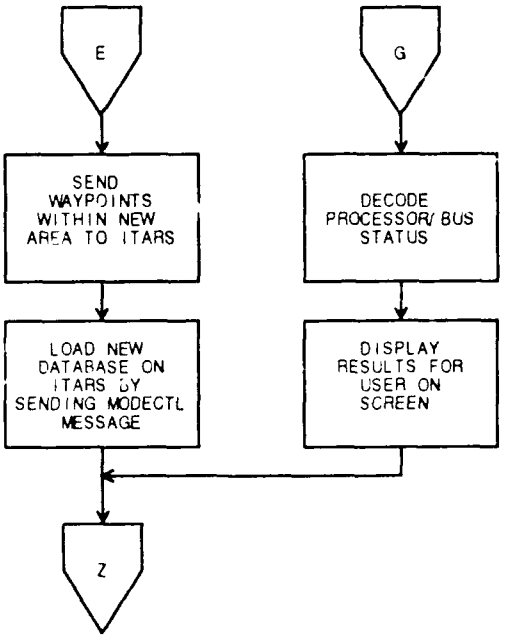



Figure 7(h)

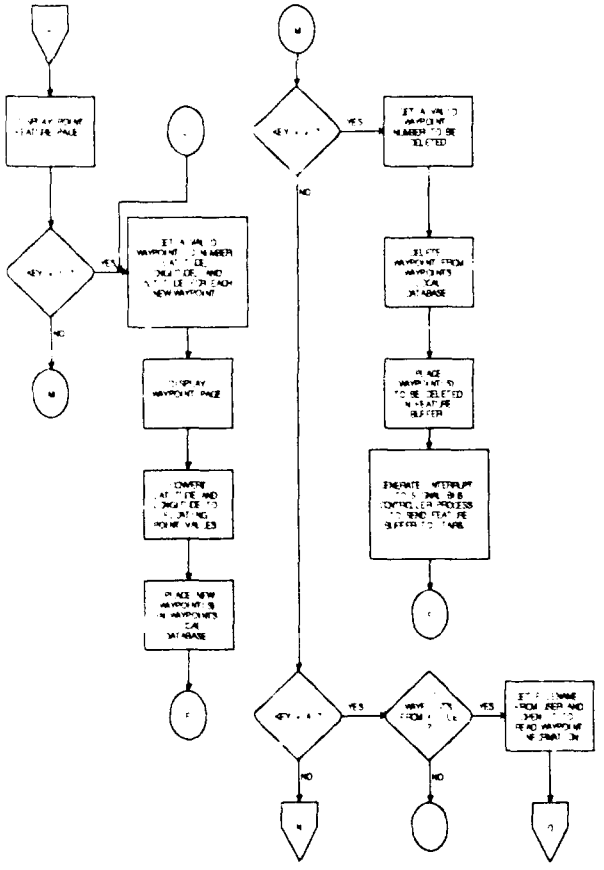


Figure 7(i)

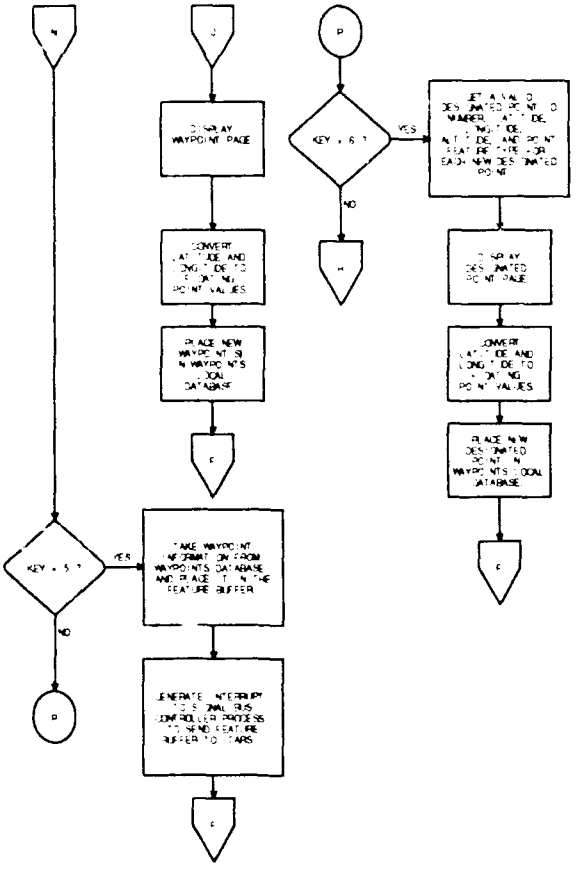


Figure 7(j)

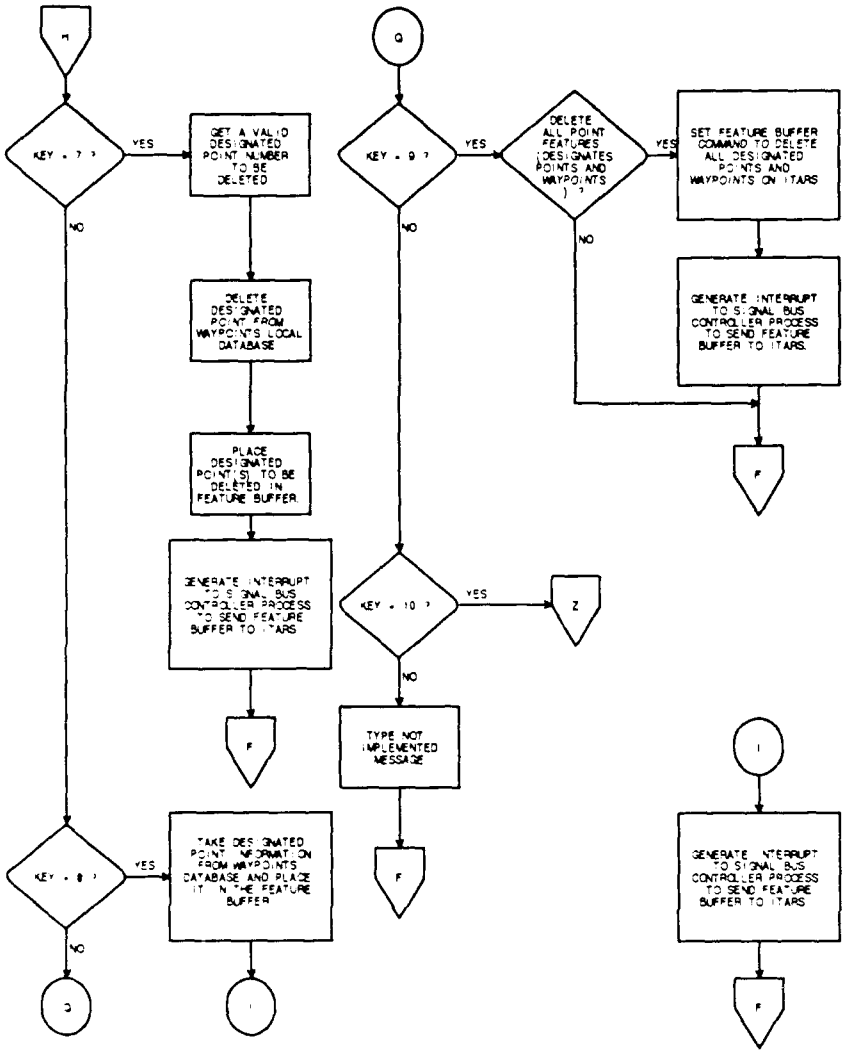



**Figure 7(k)**

ITARSSTATUS buffer and display the results on a page for the user to view. Finally, the data from the ITARSSTATUS buffer was placed in the data file for post-processing analysis.

3.2.2 Synchronous Data. The synchronous message was processed last. The synchronous message containing the aircraft state vectors was moved from the STATEVEC buffer (ITARS GCS) into the BIU memory to be sent out on the MIL-STD-1553B bus to ITARS. The data was logged into a data file for post-processing analysis. After the buffer had been sent, the Bus Controller Process waited on a reply from ITARS stating whether or not it received the buffer correctly.

3.3 User Interface Process.

3.3.1 Organization/Overview. The User Interface (UI) Process contained the WAYPOINTS local database along with mappings to the ITARS, EMITTER, and GAMING AREA Global Common Sections (GCS). The UI Process was required to draw and keep track of the Integrated Multi-function Keyboard (IMFK) pages on the VT220 terminal. The IMFK pages were the interface between the user and the GCS. The menu hierarchy and functionality were designed as in figure 6. For a flowchart of the User Interface Process, see figure 7.

3.3.2 Data Storage. Three local databases and a tree hierarchy were designed to manage the menu system for the Early Bird project.

3.3.2.1 STATIC Database. The local STATIC database was defined to be a two-dimensional character string array that contained all the static lines displayed on a particular IMFK page (see figure 8). The contents of each record within the database held the static text to be displayed on the IMFK at that particular key location. This two-dimensional array was laid out with the page number as the row index and the key number as the column index. Since there were 5 defined pages (ITARS, PLAN PAGE, PERSP PAGE, SYSTEM HEALTH, and POINT FTR PAGE) and there were 10

keys plus a title line for each page (column 0), this array or database was 5 X 11. For example, from figure 8, on the PT FTR page at key location 3, the text would read WPT SEQ according to record (5,3) of the database.

3.3.2.2 KEY_VALUES Database. The KEY_VALUES database was designed to contain the values corresponding to the data values being displayed for the depiction (shaded, contour, clearance plane, contour/shaded relief), range (5,10,20,40), orientation (North Up, Track Up), centering (centered, decentered), declutter, lights, and thinning options. This two-dimensional integer array was designed with the page number as the row index and the key number (1-10) as the column index (see figure 9). The contents of each record contained either the value to be displayed on the IMFK or a bit pattern that mapped into a character string to be displayed. The mapping functions for the keys are displayed underneath the database layout. For example, under ORIENTATION, it is mapped onto keys three and eight on the PLAN PG and only key eight for the PERSP PG. The mapping was accomplished by the software as follows: if the PLAN PG, key three contains a one as it does in figure 9, then according to the mapping functions given in the figure, the orientation of that display should be 'Track Up'. Since this was designed as an integer database, so that it did not have to contain integer and character elements, a mapping between integers and text was necessary. For example, if KEY_VALUES (2,2) contains a zero, then the ITARS multimode display on the plan page, should be in the shaded mode. When the ITARS, SYS HLTH, and PT FTR pages were brought up, they did not contain any variable text which needed to be tracked. For this reason, records of those pages in the database contain a -1, signifying this record was not used. On the PLAN and PERSP PGs, though, when the user toggled through the values for a key, the value was changed in the KEY_VALUES database accordingly.

3.3.2.3 VAR Database. The VAR database is also a two-dimensional array that contains 10-character-long variable words. This database contains the words that were displayed on the second line of each key on an IMFK page. Again, this two-dimensional array had as its row index the page, and the column index was the key number. This database was exactly like the KEY_VALUES database, except it held text, not integer values. For example from figure 10, if VAR(2,2) was set to match KEY_VALUES (2,2) from above, then VAR (2,2) should contain 'SHADED RELIEF'. The VAR and KEY_VALUES databases must have the same values and words corresponding, or the displays on ITARS will not be correct. The use of the VAR database allowed the program to keep track of the second text line to be displayed for an IMFK key while the KEY_VALUES database contained integer values which represented what mode the actual ITARS displays were in. These two databases could have been implemented as one, but comparisons are quicker with integer values (instead of character strings), the integer values were quickly and easily changed in the KEY_VALUES database, and there was less correlation to be done between the two databases. The use of one database would have consumed more CPU time consuming, and been more confusing.

3.3.3 Page/Menu Operation. After initialization of the databases and variables was complete, the UI Process began by clearing the VT220 Screen and displaying the first page of the IMFK, called ITARS. When the user selected a key from this page, another page from the ITARS tree hierarchy was brought up or the user was prompted for data entry as required. If the user selected an undefined key, the center, bottom of the

IMFK page would display 'KEY NOT IMPLEMENTED'. The key was selected by the user typing in the key number which was displayed in the lower left corner of the VT220 at the : prompt (see figure 11).

3.3.3.1 ITARS Page. The ITARS page options for keys one and two, when selected, redrew the IMFK screen and brought up the PLAN PAGE and PERSP PAGE menus, respectively. If key five was selected, the user was prompted for the number of the new gaming area database that was to be loaded from the ITARS optical disk. While the database was being loaded, the message 'LOADING NEW DATABASE' was displayed. If key six was selected, the POINT FEATURE PAGE was displayed, or if key nine was selected, ITARS was required to send its system health to the UI Process. This was accomplished by the User Interface Process generating an interrupt to the Bus Controller, requiring it to request ITARS to send its system health. The Bus Controller Process sent a 1553 message to ITARS and waited on the BIU to interrupt when it received a reply from ITARS. At this time, the Bus Controller placed the system health message into the ITARS GCS and generated an interrupt for the UI Process. The UI Process upon reception of the interrupt, decoded the message and displayed the status of the ITARS processors on the SYSTEM HEALTH page. If the user selected key 10, a prompt was sent to the user to make sure the user really wanted to exit the program before the process was halted. Keys three, four, seven, and eight were not implemented.

3.3.3.2 PLAN Page. From the ITARS page, key one brought up the PLAN PAGE as shown in figure 12. This page was laid out so that keys one-five changed the moding of the multimode channel display of ITARS (raster 2), while keys six-nine changed the moding of the dedicated plan view channel (raster 1). The RANGE values were 5, 10, 20, and 40 nautical miles for both the multimode and dedicated plan view channels. The DEPICTION values were SHADED, CONTOUR, CLEARANCE PLANE, and CONTOUR/SHADED RELIEF for both display channels. The ORIENTATION of the displays was either TRACK UP or NORTH UP, and the CENTERING option either had the plane located in the center of the display area, or at the bottom of the display. The thinning option for the multimode channel, when selected, caused ITARS to display only the highest priority point features as defined by the ITARS software. Key 10 allowed the user to return back to the ITARS page. The keys on the PLAN PAGE operated as either a rotary type or toggle type. The RANGE and DEPICTION values were rotated until the user choose the values he was searching for. The ORIENTATION, CENTERING and THINNING keys were implemented as toggle switches which cycled between the two values available for the key. The user was able to view the key values on the IMFK page as he was changing them. As the user was changing the key values, he could also see view the changes being made to the display(s) on the ITARS unit simultaneously.

3.3.3.3 PERSP Page. The second key on the ITARS page brought up the PERSP PAGE as displayed in figure 13. This page was laid out exactly the same as the PLAN PAGE, so that keys one thru five changed the moding of the multimode channel, while keys six thru nine changed the moding of the dedicated plan view channel. The RANGE values were 5, 10, 20, and 40 for the dedicated plan view channel. The DEPICTION values were SHADED, CONTOUR, CLEARANCE PLANE, and CONTOUR/SHADED RELIEF for the dedicated plan view channel, while the values were SHADED RELIEF (HUD RIDGELINE), RIDGE, SLICE (HUD OFF) and PROFILE for the multimode display channel. The ORIENTATION of the dedicated plan view

channel was either in the TRACK UP or NORTH UP mode, and the CENTERING option either had the plane located in the center of the display area, or at the bottom of the display. The DECLUTTER option for the multimode channel allowed the user to turn off point features within a 20 nm range. Key 10 allowed the user to return back up to the ITARS page. Keys one, four, and five of the PERSP PAGE were left undefined for this application even though key five has NEXT displayed by it on the PERSP page layout in figure 13. The keys on the PERSP PAGE operated as either a rotary type or toggle type like the keys that were implemented on the PLAN PAGE. Again the user was able to view the text change on the IMFK page as the key values were changing. As the user was changing the key values, he could also see view the changes being made to the display(s) on the ITARS unit simultaneously.

3.3.3.4 POINT FEATURE Page. The POINT FTR PAGE allowed the user to send point feature commands to the ITARS unit entered at the keyboard.

3.3.3.4.1 WAYPOINT Feature. The Early Bird team took the location and numbers of the mission waypoints before each flight test, renumbered them, and entered them into a data file on the MicroVAX. The INIT WAYPT key (key four) was used when the user wanted to read in the waypoints from a data file. It prompted the user to verify his intentions, then asked for the waypoint filename, checked to make sure the file existed and read the waypoint data from the file if it existed. If the user did not want to read a data file of waypoints, but instead wanted to enter the waypoints individually, the process jumped to key one's algorithm. Key one, ADD WAYPT allowed the user to enter waypoints individually. The user was prompted by the process for the waypoint identification number (limited to 0-31), latitude and longitude in degrees, minutes and seconds, and the altitude of the waypoint. The user could enter as many waypoints as necessary and when complete, would enter a carriage return. This would bring up page(s) that would list the waypoints values for the user to view, for verification (see figure 15). After the waypoint values were converted to the correct units, they were placed in the WAYPOINTS Common. The DEL WAYPT key, (key two) prompted the user for the number of the waypoint to be deleted. This key checked to make sure the waypoint was located in the WAYPOINTS Common and that the waypoint had been sent to ITARS. If the waypoint(s) had not been sent to ITARS an error message would be displayed for the user, suggesting that the waypoints be sent before trying to delete them. Key three is the WPT SEQ key, which was intended to be used to sequence the waypoints, but this function was not implemented within the ITARS. Key five, the SEND WAYPT key was the key required to send the data in the FEATURE buffer to the ITARS unit over 1553B. After the user either entered waypoints by hand or through a data file, or deleted a waypoint from the path, this key was required to send the data to ITARS.

3.3.3.4.2 Designated Point Features. The right side of the POINT FTR PAGE dealt with the designated points or features to be displayed on the ITARS channels. Key six was the ADD DESIG PT key. This key allowed the user to define a designated point to be displayed on ITARS. The process prompted the user for the designated point or emitter number (32-63), the latitude and longitude in degrees, minutes, and seconds, along with the altitude and point type. This key gave the user the option to test sending emitters to the ITARS unit when not monitoring the AAT databus. During the flight test, the emitters were sent across the 1553B bus from the OBMM and were sent to the ITARS unit through

the Monitor and Bus Controller Processes. Therefore, this key was not necessary during a flight test. The user could enter as many designated points as necessary and, when complete, would enter a carriage return. This would bring up page(s) that would list the designated points for the user to view, for verification (see figure 16). After the designated point values were converted to the correct units, they were placed in the EMITTER GCS. The functionality of key seven, DEL DESIG PT, is much the same as key two, the DEL WAYPT key. This key allowed the user to enter the designated point number to delete, then checked to see that the point was in the EMITTER GCS, and finally deleted that entry from the GCS. The SEND DESIG PT, key eight, forced the user to send any new designated points (or designated points to be deleted) to the ITARS unit before any other operation was preformed on the points. Key nine allowed the user to delete all the waypoints and designated points both on ITARS and within the WAYPOINTS Common and EMITTER GCS. Key 10 returned the user back to the ITARS page.

3.3.3.5 SYSTEM HEALTH Page. The SYSTEM HEALTH page (figure 17) was displayed after the UI Process received an interrupt from the Bus Controller Process confirming the ITARSSTATUS buffer had been updated. The UI Process cleared the interrupt, decoded the message, and selected the proper character string denoting if a processor board's statuses in the ITARS unit had either a 'FAIL' or 'OK' status. The user could view this page until he entered a carriage return which returned to the ITARS page.